Mobile Phone User Identification with Fuzzy Fingerprints

Nuno Homem **Joao Paulo Carvalho**

nuno.homem@hotmail.com joao.carvalho@inesc-id.pt
TULisbon – Instituto Superior Técnico
INESC-ID
R. Alves Redol 9, 1000-029 Lisboa, Portugal

Abstract[1]

Fingerprint identification is a well-known technique in forensic sciences. The basic idea of identifying a subject based on a set of features left by the subject actions or behavior can be applied to other domains. Identifying a mobile phone user based on a user fingerprint is one such application. This paper considers the problem of extracting fingerprints from call usage logs and matching them with those obtained from a set of known users. It presents an innovative fuzzy fingerprint algorithm based on vector valued fuzzy sets. Call destination numbers are used as base features to create the fingerprint. The assumption is that numbers called by each user remain approximately stable and are distinctive. The paper presents the proposed algorithm and shows some experimental results that validate the approach. The use of fast and compact algorithms is critical due to the possible huge number of users, and allows this method to be used on near real time.

Keywords: component, fuzzy fingerprints, vector valued fuzzy sets, similarity, frequent elements, approximate algorithms, data streams.

1. Introduction

In this paper, one considers the problem of extracting a fingerprint from call usage logs and then using that fingerprint to identify the user behind an unidentified phone number.

Identifying users based in their previously known usage is a known procedure in several areas such as telecommunications and financial fraud detection. For security and judicial purposes, identifying a user based on a specific usage behavior is also relevant. These detection methods are based on the fact that although a suspect or fraudster may present different identification credentials or simply no identification at all, its behavior and its relationships remain the same. Every individual keeps some relationships and habits, either personal or businesswise, stable for some time. Even if some of those relationships and habits change along time (the individual starts working in a different company, a distinct business, marries another person, etc.), some will remain stable (same family, some job, same interests). This provides the basis for the fingerprint as relationships and habits are translated into usage events. By using the destination number as a proxy for the individual behind a specific communication event, we can gather information out of a person's calls to its relationship network.

Within the scope of this paper, one will consider a mobile user session as the sequence of call records that belong to a single user, either known or unknown.

Each phone call record represents distinctive event within a session. The set of known users for whom a fingerprint is available will constitute the suspect library for unknown user sessions' identification.

Fingerprint identification is a well-known technique in forensic sciences and widely documented. In computer sciences a fingerprint is a procedure that maps an arbitrarily large data item (such as a computer file, or author set of texts) to a much compact information block, its fingerprint, that uniquely identifies the original data for all practical purposes, just as human fingerprints uniquely identify people for practical purposes.

In computer sciences, fingerprints are typically used to avoid the comparison and transmission of bulky data. For example, in order to efficiently check if a remote file has been modified, a web browser or proxy server can simply fetch its fingerprint and compare it with the fingerprint of a previously fetched copy. Fingerprints are a fast and compact way to identify items.

To serve the user identification purposes, a fingerprint must be able to capture the identity of that user. In other words, the probability of a collision, i.e., two users yielding the same fingerprint, must be small. The fingerprint must also be robust: a user should be identified even if he changes some aspects of the phone call daily use. The idea of identifying users based on a fingerprint is a very appealing one, because identification can theoretically be made on near real time.

To be useful, the fingerprint should comply with some basic criteria:

- Include a minimal set of features that describe the user (the suspect) in a compact format.
- Allow for update operations whenever new information (new logs) on the user is available.

[1] This work was in part supported by FCT (INESC-ID multi annual funding) through the PIDDAC Program funds.

- Allow for a fast comparison process once a new session owner needs to be identified.
- Scalability, i.e., performance should not degrade significantly when the number of sessions and suspects in the pool increases.
- Flexibility, i.e., should allow new suspects to be included in the process, whenever information is available.

This paper proposes a new method for identifying mobile phone users given a set of possible suspects. By using the destination number call frequencies as a proxy for the individual behind a specific session, one can gather information on the user and identify other unidentified sessions.

The first step in the proposed method is to gather the top-*k* destination number call frequencies in all known call logs of each known user. Since classical exact top-k algorithms are inefficient and require the full list of distinct elements to be kept, an approximated algorithm is used for this purpose. The Filtered Space-Saving algorithm [6][7] is used in this approach since it provides a fast and compact approximate answer to the top-*k*. This paper defends that the algorithm approximation is not an issue, as a degree of change or randomness has to be expected and incorporated into the detection method.

Once the top-k destination number call frequencies are available, the fingerprint is constructed by applying a fuzzifying function to each frequency. This paper proposes the innovative method of fuzzifying the set of features based on their order on the top-*k* list instead of their frequency value.

The last part in the process is to perform the same calculations for the session being identified and then to compare this fuzzy fingerprint with all the available users fuzzy fingerprints, i.e., the suspects' library. The most similar fingerprint is chosen and the session is assigned to the fingerprint author.

2. Relation with previous work

2.1. Signatures and fingerprints

Boltan and Hand [1] and Phua and al. [12] survey multiple methods for fraud detection using statistical and data-mining techniques but no mention is made of research aiming to identify individual fraudsters based on their behavior. A more common, although distinct, approach is the use of usage signatures to detect behavior changes as in [3], [4] and [5]. Signatures are used to monitor the individual activity and detect changes in that activity, or to compare against known fraud signatures to detect similarity of patterns, not the individuals behind the fraud. In [3] and [5] signatures are created extracting relevant features out of the individual usage, such as number of local, national and international calls, etc., based on pre-defined types of traffic classifications, and then are summarized. In [5] a list of the top-k countries and destinations called, and a count of calls that do not go to any of the top-k is used but with the purpose of detecting changes in the usage.

A recent work by Cormode et al. [1] presents the concept of signature algorithms applied to iterations between individuals and provides some signature schemes to network traffic and telecommunication calls. A generic approach and a theoretical framework for signatures communication graphs analysis are provided. The "signatures" concept has some common points to the proposed user fingerprint. However, one prefers to use the "fingerprint" designation, as the algorithm aims at extracting information about the user that he has not provided knowingly, while a "signature" usually refers to information that was created specifically to identify someone or something. In [1] the feature extraction is exact; the use of approximated algorithms is suggested and several distinct distances are proposed.

2.2. Fuzzy Signatures

The fuzzy fingerprint concept is a generalization of the Vector Valued Fuzzy Sets (VVFS) concept introduced by Kóczy [8]. The qualitative meaning of an object is represented by the quantities of the VVFS.

The vector valued fuzzy sets concept has also been used in [9] to introduce the fuzzy signature concept. Fuzzy signatures can model sparse and hierarchically correlated data with the help of hierarchically structured VVFS and a set of not-necessarily homogenous and hierarchically organized aggregation functions.

3. The Filtered Space-Saving Algorithm

To allow the use of user identification techniques in near real time and for a large number of potential suspects and sessions, a key issue is to be able to extract the relevant features using an efficient algorithm with reduced memory usage. In this case, features are the most frequently called phone numbers for the user. The choice was to use an approximate top-*k* algorithm capable of generating good quality estimates using a reduced memory footprint. The Filtered Space-Saving algorithm [7] was chosen. Filtered Space-Saving, originally presented in [6], is an evolution from Space-Saving algorithm presented by Metwally and al. [10].

The Filtered Space-Saving (FSS) algorithm uses a bitmap counter with *h* cells, each containing two values, α_i and c_i, standing for the error and the number of monitored elements in cell *i*. An hash function that transforms the input values (words) into an uniformly distributed integer range is used to obtain *h(x)*. The hashed value *hash(x)* is then used to increment the corresponding cell on the bitmap counter. Initially all values of α_i and c_i are set to 0.

The second storage element is a list of monitored elements *A* with size *m*. The list is initially empty. Each element contains three parts; the value itself v_j, the estimate count f_j and the associated error e_j.

The minimum required value to be included in the monitored list is always the minimum of the estimate counts, *μ= min {f_j}*. While the list has free elements, the minimum is set to 0.

The algorithm is quite simple. When a new element is received, its hash is calculated and the bitmap counter is checked. If there are already monitored elements with that same hash *($c_i > 0$)* the list is searched to see if this particular element is already there. If the element is in the list then the estimate count f_j is incremented. If the element is not in the list then it is checked to see if it should be added.

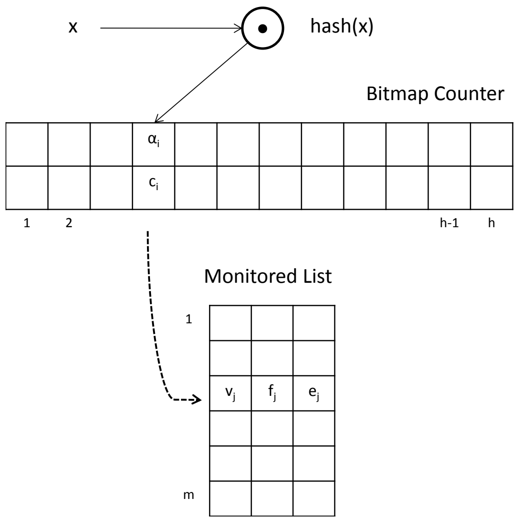

Fig. 1: FSS Algorithm Diagram.

A new element will be inserted into the list if $\alpha_i + 1 \geq \mu$. If the element is not monitored then α_i is incremented. In fact this α_i stands for the number of elements with hash value i that have not been counted in the monitored list; it is the maximum number of times an element that is not in the list and that has this hash value could have been observed.

```
Algorithm: FSS(h cells, m counters, S stream)

begin
for each element, x, with value w, in S {
    set min to min {$f_j$}
    let i be the hash(x) mod h
    if $c_i$ is not 0 {
        if x is monitored {
            let j be the index of x in the list
            increment $f_j$
            continue for next x
        }
    } // this will only be executed if x is not monitored

    if $\alpha_i$ + 1 >= min {
        if list size equals m {
            let m be the index with lower $f_j$
            and for same $f_j$ with higher $e_j$
            let k be the hash(x) mod h
            decrement $c_k$
            set $\alpha_k$ = $f_i$
            remove $v_m$
        }
        include x in the list in index j
        set $v_j$ to x
        set $e_j$ to $\alpha_i$ and $f_j$ to $\alpha_i$+1
        increment counter $c_i$
    } else {
        increment $\alpha_i$
    }
}// end for
end
```

Fig. 2: The FSS Algorithm

4. The Fuzzy Fingerprint Algorithm

The main concept behind this algorithm is that users have a stable enough behavior that allows a set of features to be extracted, fuzzified and then compared. The most frequent phone call destination numbers of a single user present the required stability.

User phone usage can be analyzed as a set of calls generated by a stable distribution that depends only on the user. Only counts for each distinct destination number are in fact relevant and are used as variables.

The set of distinct destination numbers to consider in the fingerprint should be large enough to allow a comprehensive sample of the user behavior along time. The FSS algorithm requires two parameters, the size of the monitored list *m* and the size of the bitmap counter *h*. In all the tests the parameters were set proportional to the number of distinct destination numbers used in the fingerprint: *m = 3k, h = 9k*.

4.1. Fuzzy Fingerprint Creation

The full set of known sessions are processed through the modified FSS algorithm to compute the approximated top-*k* list of destination numbers and frequencies for each user. Consider T_j is the set of logs by the user *j*. The result consists of a list of *k* tuplets *{v_i, n_i}* where v_i is the *i*-th most frequent called number and n_i the corresponding count estimate.

To create the actual fingerprint, the top-*k* list has to be fuzzified. The choice of the fuzzifying function is critical and the chosen approach is to assign a membership value to each destination number in the set based only on the order in the list. In fact, experiments have shown that the order of the frequency seems more relevant than its actual value. The more frequent destination numbers will have a higher membership value.

Several alternative membership attribution functions for each element *i* of the top-*k* list are tested in this paper. The simplest one is:

$$\mu_{flat}(i) = \frac{k-i}{k} \quad (1)$$

Function μ_{par} is inspired in the Pareto rule – 80% of the membership value is assigned to first 20% elements in the ranking:

$$\mu_{par}(i) = \begin{cases} 1-(1-b)\frac{i}{k} & if\ i \quad a \\ a\left(1-\frac{i-a}{k-a}\right) & if\ i \geq a \end{cases} \quad (2)$$

The third function is μ_{erfc}, based on the complementary error function:

$$\mu_{erfc}(i) = 1 - erf\left(\frac{2i}{k}\right), \quad (3)$$

where *erf()* is the Gauss error function. Figure 3 presents the used functions.

The fingerprint based on the top-*k* list (size-*k* fingerprint, *Φ*), consists on a size-*k* fuzzy vector where each position *i* contains a element v_i and a value μ_i representing the fuzzified value of v_i's rank (the membership of the rank).

An user *j* will be represented by its fingerprint $\Phi_j = \Phi(T_j)$. Formally, fingerprint $\Phi_j = \{(v_{ji}, \mu_{ji}) | i = 1..k_j\}$ has length k_j; let $S_j = \{v_{ji} | i = 1..k_j\}$ be the set of *v*'s in Φ_j.The set of all user fingerprints will constitute the fingerprint library.

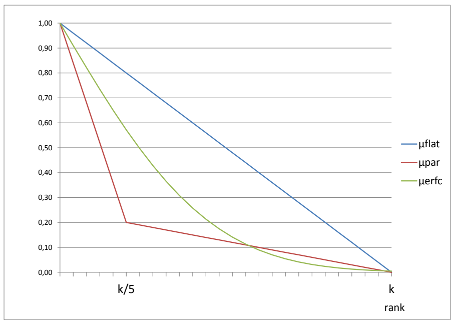

Fig. 3: Fuzzyfying functions

4.2. Fuzzy Fingerprint Detection

In order to find the user of an unknown session *L*, one starts by computing the size-*k* fingerprint of *L*, Φ_L. Then one compares the fingerprint of *L* with the fingerprints Φ_j of all users present in the fingerprint library. The unknown user is identified as user *j* if he has the most similar fingerprint to Φ_L. Fingerprint similarity, $sim(\Phi_L, \Phi_j)$, is calculated using (4):

$$sim(\Phi_L, \Phi_j) = \sum_{v \in S_L \cup S_j} \frac{\min\left(\mu_v(\Phi_L), \mu_v(\Phi_j)\right)}{k}, \qquad (4)$$

where $\mu_v(\Phi_x)$ is the membership value associated with the rank of element *v* in fingerprint *x*. This similarity function is based on the minimum or Gōdel t-norm, but other t-norms could also be used.

5. Experimental Results

5.1. Experimental data set

Experiments were performed using a data set consisting of five weeks of call logs from randomly selected customers of a large mobile telecommunications company. The data is annonymized but every user and number in the call record are identified by a unique tag.

The logs from the four initial weeks were used to create the suspect library and constitute the training set. The test set was created with the sessions from the remaining week of call logs. A total of 1 615 480 call records were included in the training set, and 426 861 call records in the test set.

A total of 2270 users were considered for these tests, each with a minimum 240 records in the training set and 60 records in the test set. Note that this is a small number of call records; many of the users generate more than a couple of hundred records per week.

Although the sample period is not long, it is interesting to see the distinct called number distribution per user. Fig. 4 presents the percentage of calls for each of the most frequent numbers called by the user. On average 80% of the users calls are made to just 34 numbers. Fig. 4 also presents the distribution of consecutive call destinations, and the mixed of the two distributions. Consecutive call destinations are relevant because one user may call one number and then another causally related number. This path may be distinct from other user paths and helps distinguising the users.

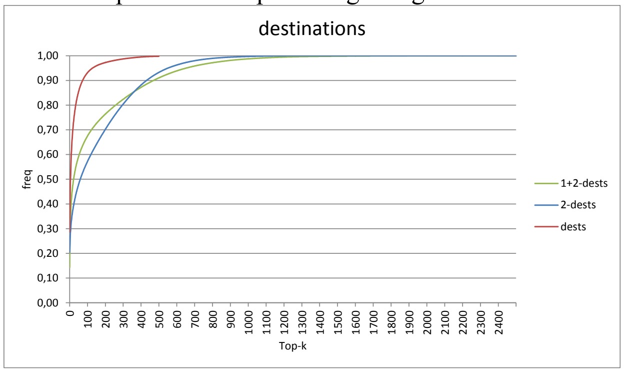

Fig. 4: Destination number distribution

5.2. Test methodology

The distinct fuzzifying functions were tested for each of the available users and sessions. Each test returns a ranked list of the suspects. A test is considered to be positive if the real user is returned in the first position of the ranked list. Precision measures the ratio of positive tests for each of the function. Tests were performed for multiple values of *k* and for each of the tested algorithms.

To evaluate the precision of the proposed algorithms against crisp signature algorithms, one will present experimental results obtained using equivalent methods proposed in [2], where a variety of distance functions were presented to compare signatures. They are generalized from known measures, and take into account both set overlap as well as weighted occurrence. Signatures are k sized tuples containing an element v_i and a value w_i representing the frequency of the element. Each signature $\Omega_i = \{(v_{ij}, w_{ij}) | j = 1..k_i\}$ has length k_i; let $S_i = \{v_{ij} | j = 1..k_i\}$ be the set of *v*'s in Ω_i. The considered distances were adapted from [2]:

$$D_{jac}(\Omega_1, \Omega_2) = 1 - \frac{S_1 \cap S_2}{S_1 \cup S_2} \qquad (5)$$

$$D_{dice}(\Omega_1, \Omega_2) = 1 - \frac{\sum_{v \in S_1 \cap S_2} \min(w_1, w_2)}{\sum_{v \in S_1 \cup S_2} \max(w_1, w_2)} \qquad (6)$$

$$D_{hel}(\Omega_1, \Omega_2) = 1 - \frac{\sum_{v \in S_1 \cap S_2} \sqrt{w_1 \cdot w_2}}{\sum_{v \in S_1 \cup S_2} \max(w_1, w_2)} \tag{7}$$

It is easy to verify that all these distance functions yield values in [0, 1]. D_{jac} is based on Jaccard coefficient, where the node weights are not taken into account; it is minimized when S1 = S2, and it equals 1 when their overlap is empty. D_{dice} is a scaled version of the distance based on Dice criterion, which factors in node weights; it gives an added premium if the individual weights in S_1 and S_2 are similar. D_{hel} is based on Hellinger distance .

ROC curves are presented for every algorithm in order to better characterize the classifier response. ROC curves are a standard measure in statistics [10]. ROC curves map the ratio of false positives against the ratio of true positives, with the false positives in the x-axis and true positives in the y-axis. ROC curves were constructed by first running every test sample against the suspect library and obtaining a ranked list. Let T be the number of tests and U the number of suspects. The total number of elements in the answer lists is *TU*. The total number of positive tests is *T* (there is always one correct user for each unknown session) and the number of incorrect tests is *T(U-1)*. The ROC curve starts at the origin (0, 0). For each rank, in ascending order, the number of positive tests *p* with that rank are used to draw a vertical line upwards by *p/T*. For the same rank a horizontal line is draw to the right by *(T-p)/(T(U-1))*.

The Area Under the ROC Curve (AUC) [10], a standard measure in statistics, was also computed. AUC is an indicator of the overall precision of the classifier: if the AUC is 0.5, the classifier is no better than random selection; higher values indicate better precision. An AUC of 1 indicates a perfect classifier.

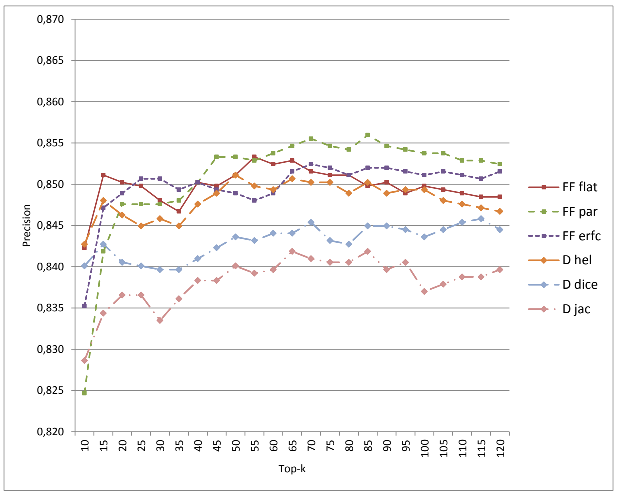

Fig: 5: Algorithms precision for single destination number

5.3. Single destination number fuzzy fingerprints

The first set of experiments uses single destination number fingerprints. Fig: 5 presents the precision of fuzzy fingerprints using the three fuzzifying functions described and the results obtained using the three crisp distance functions. The difference between fuzzy and crisp algorithms is very significant. The fuzzy fingerprints deal better with the very sparse nature of the call destinations and significant differences of use in the training and test periods. Differences between the three proposed fuzzifying functions are not very relevant: the Pareto based function achieves the best results, followed closely by the complementary gaussian error function. These functions weight more the most frequent elements.

Fig: 5 presents the ROC curve for each of the tested algorithms. To better show the differences Fig. 6 presents the zoom of the most relevant area of the ROC.

5.4. Consecutive destination numbers fuzzy fingerprints

The second set of experiments uses pairs of consecutive destination numbers fingerprints, i.e. the pair of destination numbers of two consecutive calls. Fig. 8 presents precision for each of the algorithms. Both crisp and fuzzy algorithms present slighty worse results, probably because the extra information contained in the order of the accesses does not compensate for the extra noise, and the results can only be achieved at the expense of much higher number of top-*k* elements. Fuzzy algorithms perform much better than crisp algorithms but is interesting to see that the worst performing algorithm in the single destination number tests, the Jaccard based distance performes much better with consecutive destination numbers. Fuzzy fingerprints using the flat function achieves the best results, followed closely by the complementary gaussian error function. Fig. 9 and Fig. 10 present the ROC curves.

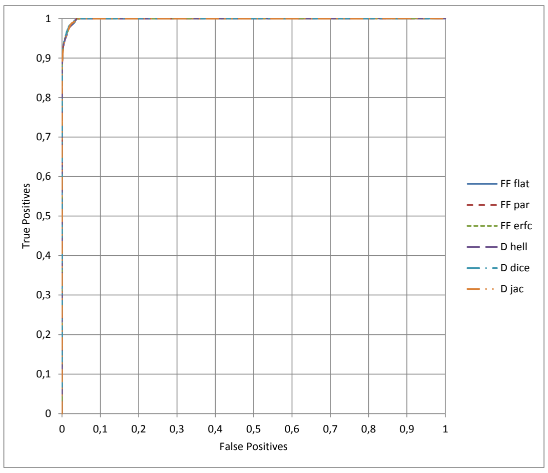

Fig. 6: Algorithms ROC for single destination number

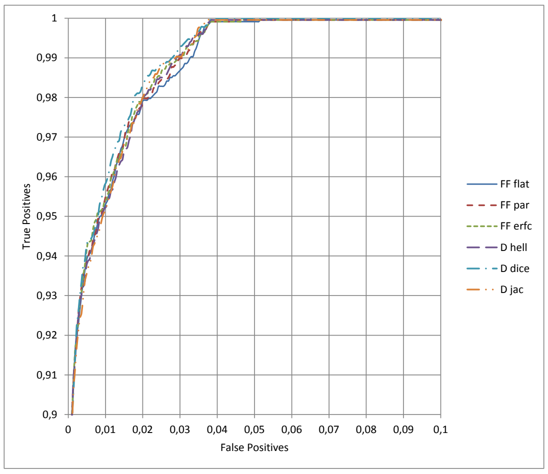

Fig. 7: Detailed ROC for single destination number

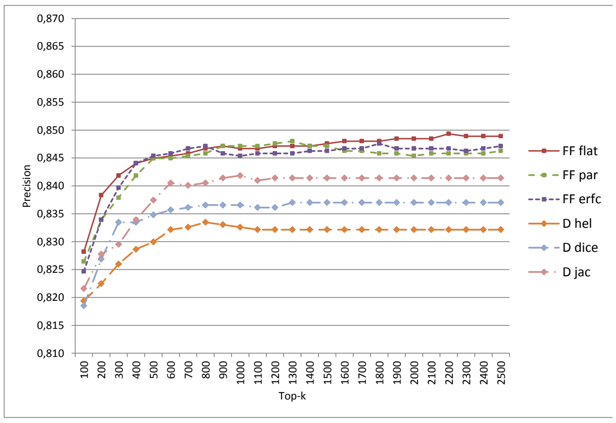

Fig. 8: Algorithms precision for consecutive destination numbers

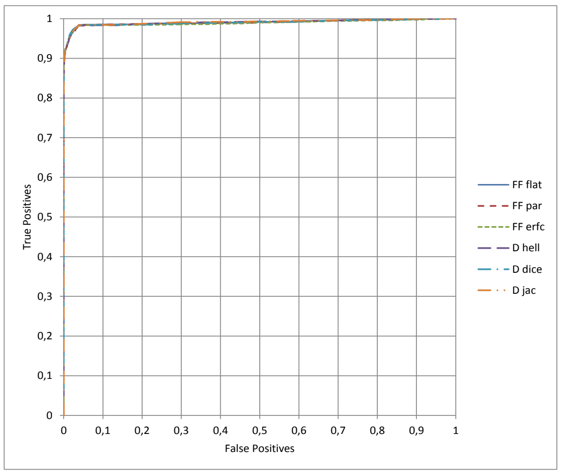

Fig. 9: Algorithms ROC for consecutive destination numbers

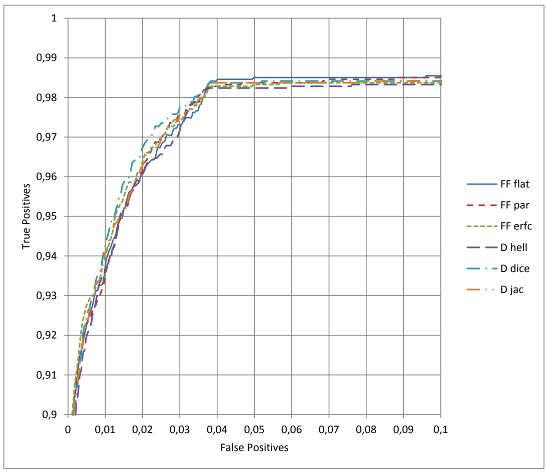

Fig. 10: Detailed ROC for consecutive destination numbers

5.5. Mixed fuzzy fingerprints

The final set of experiments use a mix of both the single destination and consecutive destinations fingerprints. Both elements are mixed in a single top-k list. Fig. 11 presents the precision for each of the algorithms. Fig. 12 and Fig. 13 present the corresponding ROC curves.

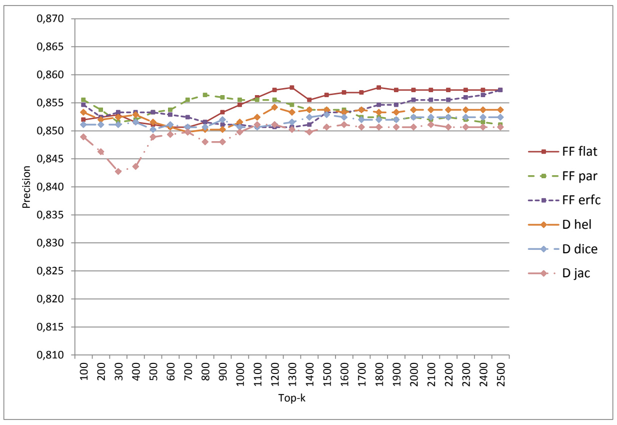

Fig. 11: Algorithms precision for mixed fingerprints

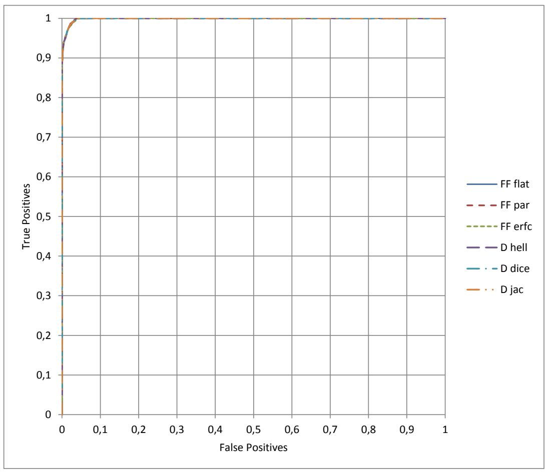

Fig. 12: Algorithms ROC for mixed fingerprints

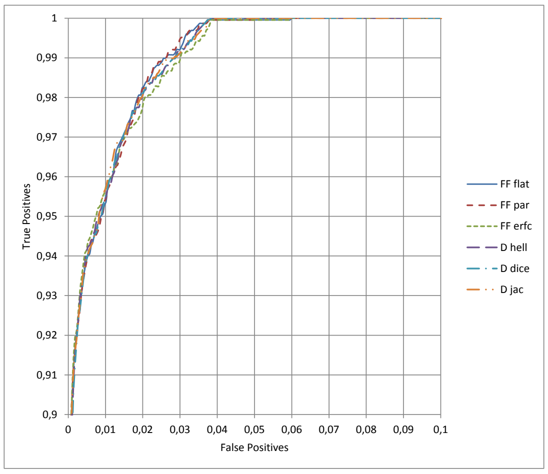

Fig. 13: Detailed ROC for mixed fingerprints

There is a slight increase in performance of both fuzzy and crisp algorithms. The most relevant beneficit for the use of the mixed fingerprint is that performance remains stable for a wider range of *k*, although *k* values are higher than for single destination number tests. Table 1 presents the peak performance values for all of the experiments and algorithms.

6. Conclusions and Future Work

This work shows how fuzzy methods may be used to identify the users behind an unidentified mobile phone session. The use of simple fuzzy techniques based on approximate algorithms leads to very interesting results.

The fact that the results remain stable for a wide range of values of *k* and for several fuzzification functions shows that the proposed method is robust.

The large number of suspects it supports and the ability to include new suspects or update existing user fuzzy fingerprints is critical to the use of the method in long-term detection processes. New fuzzy fingerprints can be created and added to the suspect's library at any time, and new session logs from known users can be added to the fuzzy fingerprint. Update to one fuzzy fingerprint does not influence all the others.

The use of order information, by using consecutive events as base features, enriches the fingerprints, even with such sparse data as phone destination numbers. One can expect that this approach could lead to even better improvements in domains where event order is more relevant.

The proposed method should not be seen as a specific method; it can be used in other domains to identify individual behavior based on events. The proposed method will identify individuals as long as they present a stable event distribution. Future work should also test the inclusion of additional features available in the call logs.

		Fuzzy Fingerprints			**Crisp Distances**		
		flat	***par***	***erfc***	***hell***	***dice***	***jac***
Tests		2270	2270	2270	2270	2270	2270
Authors		2270	2270	2270	2270	2270	2270
Dests	**Peak k**	55	85	70	50	115	65
	Peak Accuracy	0,8533	0,8559	0,8524	0,8511	0,8458	0,8419
	AUC	0,9984	0,9984	0,9987	0,9986	0,9989	0,9984
2-Dests	**Peak k**	2200	1300	1800	800	1300	1000
	Peak Accuracy	0,8493	0,8480	0,8476	0,8335	0,8370	0,8419
	AUC	0,9911	0,9906	0,9893	0,9912	0,9907	0,9917
1+2-Dests	**Peak k**	1300	800	2500	1200	1500	1100
	Peak Accuracy	0,8577	0,8564	0,8573	0,8542	0,8529	0,8511
	AUC	0,9993	0,9993	0,9992	0,9993	0,9992	0,9993

Table 1: Results for experiments with destination numbers

References

[1] R. Bolton and D. Hand, *Statistical Fraud Detection: A Review*, Statistical Science, Vol. 17, No. 3, 235–255, 2002.

[2] G. Cormode, F. Korn, S. Muthukrishnan, Yihua Wu, *On signatures for communication graphs*, IEEE 24th International Conference on Data Engineering (ICDE), 2008.

[3] C. Cortes and D. Pregibon, *Signature-Based Methods for Data Streams*. Data Mining and Knowledge Discovery 5: 167-182, 2001.

[4] T. Fawcett, and F. Provost, *Activity monitoring: Noticing Interesting Changes in Behavior*. Proc. of SIGKDD99, 53-62, 1999.

[5] P. Ferreira, R. Alves, O. Belo and L. Cortesão, *Establishing Fraud Detection Patterns Based on Signatures*, Proceedings of the 6th Industrial Conference on Data Mining, ICDM2006, Lecture Notes in Computer Science, Springer, 2006.

[6] N. Homem and J. Carvalho, *Estimating Top-k Destinations in Data Streams*, Computational Intelligence for Knowledge-Based Systems Design, Springer Berlin / Heidelberg, pages 290-299, 2010.

[7] N. Homem and J. Carvalho, *Finding top-k elements in data streams*, Information Sciences, 180(24), pp. 4958-4974, Dec. 2010, Elsevier.

[8] L. Kóczy, *Vector valued fuzzy sets*, BUSEFAL-Bulletin for Studies and Exchanges on Fuzziness and its Applications, pages 41–57, 1980.

[9] L. Kóczy, T. Vámos, G. Biró, *Fuzzy signatures*, in: EUROFUSE-SIC99, 1999.

[10] S. Mason and N. Graham, *Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation*. Q. J. R. Meteorol. Soc, 30:291–303, 1982.

[11] A. Metwally, D. Agrawal and A. Abbadi, *Efficient Computation of Frequent and Top- k Elements in Data Streams*, Technical Report 2005-23, University of California, Santa Barbara, September 2005.

[12] C. Phua, V. Lee, K. Smith, and R. Gayler, "A Comprehensive Survey of Data Mining-Based Fraud Detection Research", http://www.bsys.monash.edu.au/people/cphua/, Mar. 2007.